WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, March 16, 2005

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ		
☐	L11	L10 and L3	10
☐	L10	L7	41
	DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L9	L8 and network$	8
☐	L8	L5	72
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L7	L6 and stack	41
☐	L6	L5 and thread	155
☐	L5	(lock$ or unlock$) near3 contention	412
	DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ		
☐	L4	L3 or 718/106-108.ccls.	15839
☐	L3	L2 or 709/227-229.ccls.	14734
☐	L2	L1 or 707/7-10.ccls.	9754
☐	L1	717/100-123,126-127.ccls.	3306

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, March 16, 2005

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L17	L16 and thread	57
☐	L16	L10 and stack	81
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L15	L12 and ((light or heavy) near object)	7
☐	L14	L12 and L4	60
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L13	L12	0
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L12	L10 and operating	156
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L11	L10	6
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L10	L9 or L8	199
☐	L9	L7 and (unlock$ near object)	54
☐	L8	L7 and (lock$ near object)	191
☐	L7	content service or content server or contention	29132
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L6	content service or content server or contention	21882
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L5	content service or content server	10476
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L4	L3 or L2 or L1	12908
☐	L3	(718/102 |718/103 |718/104 |718/105 |718/106 |718/107 |718/108).ccls.	3353
☐	L2	(707/7 |707/8 |707/9 |707/10).ccls.	6506
☐	L1	(717/100 |717/101 |717/102 |717/103 |717/104 |717/105 |717/106 |717/107 |717/108 |717/109 |717/110 |717/111 |717/112 |717/113 |717/114 |717/115 |717/116 |717/117 |717/118 |717/119 |717/120 |717/121 |717/122 |717/123 |717/162 |717/163 |717/164 |717/165 |717/166 |717/167).ccls.	3316

END OF SEARCH HISTORY

Terms used
lock unlock contention thread stack object network

Found **56** of **151,219**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 56 Result page: **1** 2 3 next

Relevance scale ☐☐☐■■

1 An efficient meta-lock for implementing ubiquitous synchronization

Ole Agesen, David Detlefs, Alex Garthwaite, Ross Knippel, Y. S. Ramakrishna, Derek White

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Programs written in concurrent object-oriented languages, especially ones that employ thread-safe reusable class libraries, can execute synchronization operations (lock, notify, etc.) at an amazing rate. Unless implemented with utmost care, synchronization can become a performance bottleneck. Furthermore, in languages where every object may have its own monitor, per-object space overhead must be minimized. To address these concerns, we have developed a meta-lock to mediate access to synchro ...

Keywords: concurrent threads, object-oriented language implementation, synchronization

2 The performance implications of thread management alternatives for shared-memory multiprocessors

T. E. Anderson, D. D. Lazowska, H. M. Levy

April 1989 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1989 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 17 Issue 1

Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, references, citings, index terms

Threads ("lightweight" processes) have become a common element of new languages and operating systems. This paper examines the performance implications of several data structure and algorithm alternatives for thread management in shared-memory multiprocessors. Both experimental measurements and analytical model projections are presented. For applications with fine-grained parallelism, small differences in thread management are shown to have significant performance imp ...

3 Vertical profiling: understanding the behavior of object-priented applications

Matthias Hauswirth, Peter F. Sweeney, Amer Diwan, Michael Hind

October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and**

applications, Volume 39 Issue 10

Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, references, index terms

Object-oriented programming languages provide a rich set of features that provide significant software engineering benefits. The increased productivity provided by these features comes at a justifiable cost in a more sophisticated runtime system whose responsibility is to implement these features efficiently. However, the virtualization introduced by this sophistication provides a significant challenge to understanding complete system performance, not found in traditionally compiled languages ...

Keywords: hardware performance monitors, perturbation, software performance monitors, vertical profiling, whole-system analysis

4 Implementation of Argus

B. Liskov, D. Curtis, P. Johnson, R. Scheifer

November 1987 **ACM SIGOPS Operating Systems Review , Proceedings of the eleventh ACM Symposium on Operating systems principles**, Volume 21 Issue 5

Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

Argus is a programming language and system developed to support the construction and execution of distributed programs. This paper describes the implementation of Argus, with particular emphasis on the way we implement atomic actions, because this is where Argus differs most from other implemented systems. The paper also discusses the performance of Argus. The cost of actions is quite reasonable, indicating that action systems like Argus are practical.

5 Locking effects in multiprocessor implementations of protocols

Mats Björkman, Per Gunningberg

October 1993 **ACM SIGCOMM Computer Communication Review , Conference proceedings on Communications architectures, protocols and applications**, Volume 23 Issue 4

Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We investigate how to exploit shared memory multiprocessors for parallel protocol processing. We present a multiprocessor implementation of the x-kernel protocol environment from the University of Arizona. A "processor-per-message" paradigm is used to partition the work over processors. Locks are used to protect shared protocol state and data. Mutual exclusion by locking can be costly if the parallel protocol code frequently accesses shared state and data. This paper addresses the effect of lock ...

6 Executing Java threads in parallel in a distributed-memory environment

Mark W. MacBeth, Keith A. McGuigan, Philip J. Hatcher

November 1998 **Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(194.63 KB) Additional Information: full citation, abstract, references, citings, index terms

We present the design and initial implementation of Hyperion, an environment for the high-performance execution of Java programs. Hyperion supports high performance by utilizing a Java-bytecode-to-C translator and by supporting parallel execution via the distribution of Java threads across the multiple processors of a cluster of Linux machines. The Hyperion run-time system implements the Java memory model using an efficient communication substrate previously developed for Linux and Fast Ethernet ...

7

Techniques for obtaining high performance in Java programs
Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, David J. Lilja
September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3

Full text available: pdf(816.13 KB) Additional Information: full citation, abstract, references, citings, index terms

This survey describes research directions in techniques to improve the performance of programs written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portabili ...

Keywords: Java, Java virtual machine, bytecode-to-source translators, direct compilers, dynamic compilation, interpreters, just-in-time compilers

8 Experience Using Multiprocessor Systems—A Status Report
Anita K. Jones, Peter Schwarz
June 1980 **ACM Computing Surveys (CSUR)**, Volume 12 Issue 2

Full text available: pdf(4.48 MB) Additional Information: full citation, references, citings, index terms

9 Performance modeling of multiprocessor implementations of protocols
Mats Björkman, Per Gunningberg
June 1998 **IEEE/ACM Transactions on Networking (TON)**, Volume 6 Issue 3

Full text available: pdf(285.80 KB) Additional Information: full citation, references, citings, index terms, review

Keywords: multiprocessor, parallel communication protocols, performance modeling, queueing network model

10 Source-level global optimizations for fine-grain distributed shared memory systems
R. Veldema, R. F. H. Hofman, R. A. F. Bhoedjang, C. J. H. Jacobs, H. E. Bal
June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming**, Volume 36 Issue 7

Full text available: pdf(112.60 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes and evaluates the use of aggressive static analysis in Jackal, a fine-grain Distributed Shared Memory (DSM) system for Java. Jackal uses an optimizing, source-level compiler rather than the binary rewriting techniques employed by most other fine-grain DSM systems. Source-level analysis makes existing access-check optimizations (e.g., access-check batching) more effective and enables two novel fine-grain DSM optimizations: object-graph aggregatio ...

11 Migration: Luna: a flexible Java protection system
Chris Hawblitzel, Thorsten von Eicken
December 2002 **ACM SIGOPS Operating Systems Review**, Volume 36 Issue SI

Full text available: pdf(1.39 MB) Additional Information: full citation, abstract, references

Extensible Java systems face a difficult trade-off between sharing and protection. On one hand, Java's ability to run different protection domains in a single virtual machine enables

domains to share data easily and communicate without address space switches. On the other hand, unrestricted sharing blurs the boundaries between protection domains, making it difficult to terminate domains and enforce restrictions on resource usage. Existing solutions to these problems restrict sharing in an ad-hoc ...

12 Performance evaluation of the Orca shared-object system

Henri E. Bal, Raoul Bhoedjang, Rutger Hofman, Ceriel Jacobs, Koen Langendoen, Tim Rühl, M. Frans Kaashoek

February 1998 **ACM Transactions on Computer Systems (TOCS)**, Volume 16 Issue 1

Full text available: pdf(179.39 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Orca is a portable, object-based distributed shared memory (DSM) system. This article studies and evaluates the design choices made in the Orca system and compares Orca with other DSMs. The article gives a quantitative analysis of Orca's coherence protocol (based on write-updates with function shipping), the totally ordered group communication protocol, the strategy for object placement, and the all-software, user-space architecture. Performance measurements for 10 parallel applications ill ...

Keywords: distributed shared memory, parallel processing, portability

13 Scalable concurrent counting

Maurice Herlihy, Beng-Hong Lim, Nir Shavit

November 1995 **ACM Transactions on Computer Systems (TOCS)**, Volume 13 Issue 4

Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The notion of counting is central to a number of basic multiprocessor coordination problems, such as dynamic load balancing, barrier synchronization, and concurrent data structure design. We investigate the scalability of a variety of counting techniques for large-scale multiprocessors. We compare counting techniques based on: (1) spin locks, (2) message passing, (3) distributed queues, (4) software combining trees, and (5) counting networks. Our comparison is based on a series of simple be ...

Keywords: combining trees, counting networks

14 Waiting algorithms for synchronization in large-scale multiprocessors

Beng-Hong Lim, Anant Agarwal

August 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 3

Full text available: pdf(2.72 MB) Additional Information: full citation, abstract, references, citings, index terms

Through analysis and experiments, this paper investigates two-phase waiting algorithms to minimize the cost of waiting for synchronization in large-scale multiprocessors. In a two-phase algorithm, a thread first waits by polling a synchronization variable. If the cost of polling reaches a limit Lpoll and further waiting is necessary, the thread is blocked, incurring an additional fixed cost, B. The choice of Lpoll

Keywords: barriers, blocking, competitive analysis, locks, producer-consumer synchronization, spinning, waiting time

15 Diffracting trees

Nir Shavit, Asaph Zemach

November 1996 **ACM Transactions on Computer Systems (TOCS)**, Volume 14 Issue 4

Full text available: pdf(729.57 KB) Additional Information: full citation, abstract, references, citings, index terms

Shared counters are among the most basic coordination structures in multiprocessor conputation, with applications ranging from barrier synchronization to concurrent-data-structure design. This article introduces diffracting trees, novel data structures for share counting and load balancing in a distributed/parallel environment. Empirical evidence, collected on a simulated distributed shared-memory machine and several simulated message-passing architectures, shows that diffracting trees scal ...

Keywords: contention, counting networks, index distribution, lock free, wait free

16 Multithreading and value prediction: Speculative lock elision: enabling highly concurrent multithreaded execution

Ravi Rajwar, James R. Goodman

December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.37 MB) Publisher Site Additional Information: full citation, abstract, references, citings

Serialization of threads due to critical sections is a fundamental bottleneck to achieving high performance in multithreaded programs. Dynamically, such serialization may be unnecessary because these critical sections could have safely executed concurrently without locks. Current processors cannot fully exploit such parallelism because they do not have mechanisms to dynamically detect such false inter-thread dependences.We propose *Speculative Lock Elision (SLE),* a novel micro-architectura ...

17 The family of concurrent logic programming languages

Ehud Shapiro

September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

18 Concurrency control: methods, performance, and analysis

Alexander Thomasian

March 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 1

Full text available: pdf(427.18 KB) Additional Information: full citation, references, citings, index terms

Keywords: Markov chains, adaptive methods, concurrency control, data contention, deadlocks, flow diagrams, load control, optimistic concurrency control, queueing network models, restart-oriented locking methods, serialiazability, thrashing, two-phase locking, two-phase processing, wait depth limited methods

19 Mostly lock-free malloc

Dave Dice, Alex Garthwaite

June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management**, Volume 38 Issue 2 supplement

Full text available: pdf(609.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern multithreaded applications, such as application servers and database engines, can severely stress the performance of user-level memory allocators like the ubiquitous malloc subsystem. Such allocators can prove to be a major scalability impediment for the applications that use them, particularly for applications with large numbers of threads running on high-order multiprocessor systems.This paper introduces Multi-Processor Restartable Critical Sections, or MP-RCS. MP-RCS permits user-level ...

Keywords: affinity, locality, lock-free operations, malloc, restartable critical sections

20 A language with distributed scope

Luca Cardelli

January 1995 **Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms

Obliq is a lexically-scoped, untyped, interpreted language that supports distributed object-oriented computation. Obliq objects have state and are local to a site. Obliq computations can roam over the network, while maintaining network connections. Distributed lexical scoping is the key mechanism for managing distributed computation.

Results 1 - 20 of 56 Result page: **1** 2 3 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

Your search matched 3 of 1138071 documents.
A maximum of 500 results are displayed, 15 to a page, sorted by Relevance Descending order.

Refine This Search:
You may refine your search by editing the current search expression or enterin new one in the text box.

lock<and>thread<and>contention Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine CNF = Conference STD = Standard

---

1 Hazard pointers: safe memory reclamation for lock-free objects
Michael, M.M.;
Parallel and Distributed Systems, IEEE Transactions on , Volume: 15 , Issue: 6 , June 2004
Pages:491 - 504

[Abstract] [PDF Full-Text (1576 KB)] IEEE JNL

---

2 Practical considerations for non-blocking concurrent objects
Bershad, B.N.;
Distributed Computing Systems, 1993., Proceedings the 13th International Conference on , 25-28 May 1993
Pages:264 - 273

[Abstract] [PDF Full-Text (848 KB)] IEEE CNF

---

3 Wait-free cache-affinity thread scheduling
Debattista, K.; Vella, K.; Cordina, J.;
Software, IEE Proceedings- [see also Software Engineering, IEE Proceedings] , Volume: 150 , Issue: 2 , April 2003
Pages:137 - 146

[Abstract] [PDF Full-Text (1155 KB)] IEE JNL

Copyright © 2004 IEEE — All rights reserved

Search Results Results **1 - 10** of about **59** for **lock contention unlock stack thread connecting** - 0.04

ttp://www.watson.org/~robert/freebsd/netperf/20040806-rwatson_netperf.diff
ry to + * avoid grabbing Giant for calls we know don't need it. + */ +
ww.watson.org/~robert/freebsd/netperf/20040806-rwatson_netperf.diff - 161k - Cached - More from this site

ttp://ftp.au.debian.org/pub/solaris-patches/101318.readme
his fix will eliminate that limitation to allow usage of years 1902 and above. ... not send an NLM **UNLOCK** reques
kits ... watchdog resets with misaligned **stack** 4045229 strptime and getdate year ... cur spin lock when **thread** P
) ...
).au.debian.org/pub/solaris-patches/101318.readme - 142k - Cached - More from this site

hangeLog-2.5.32
ummary of changes from v2.5.31 to v2.5.32 ==========
davidm@tiger.hpl.hp.com> ia64: Sync up with 2.5.18. <davidm@tiger.hpl.hp.com> ia64: Fix fls() declaration so it
lined.
ww.kernel.org/pub/linux/kernel/v2.5/ChangeLog-2.5.32 - 104k - Cached - More from this site

WN: 2.5.32 long-format changelog
ponsored Link. Multiple distros, Dual Opterons, RAID, Remote console/reboot, root access, snapshot support, pc
hanks to: 2.5.32 long-format changelog
/n.net/Articles/8593 - 117k - Cached - More from this site

ttp://www.ibm.com/software/ts/tpf/maint/tpfapardata.txt
his problem was injected by PJ26864. ... for LU 6.2 SLU **thread** sessions across PU5 networks, which ... With TC
ipport applied, the connect() API ... 1 13 The cached **unlock** policer causes excessive software locking ...
ww.ibm.com/software/ts/tpf/maint/tpfapardata.txt - 191k - Cached - More from this site

ummary of Base Operating System Patches
ru64 UNIX 5.1 and TruCluster Server 5.1 Patch Summary and Release Notes for Patch Kit-0003: ... seen only wh
tack had been user defined ... pointers were connected, hereby **connecting** the ACTIVE and INACTIVE ... an ina
nlock the page. Continued attempts ...
30097.www3.hp.com/docs/patch/51/bl17a/HTML/PTCHSMCH.HTM - 96k - Cached - More from this site

ummary of Base Operating System Patches
ru64 UNIX 5.1 and TruCluster Server 5.1 Patch Summary and Release Notes for Patch Kit-0004: ... seen only wh
tack had been user defined ... is used on a **thread** that is marked as blocked ... systems in which the **unlock** dis
e ...
30097.www3.hp.com/docs/patch/51/bl18/UNTITLED/PTCHSMCH.HTM - 139k - Cached - More from this site

inux 2.5.32 - koko ChangeLog
ummary of changes from v2.5.31 to v2.5.32 ========== ia64: Syn
64: Fix fls() declaration so it actually gets inlined.
nnikala.net/2002/35/linux2532fullchangelog.php - 107k - Cached - More from this site

IX Maintenance Package for AIX 4.3.3
Reduce VMM_lock_lock **lock contention** IY03977 3 ATM requests ... IY04386 2 program exception - **stack** poi
'04388 3 ... working IY05182 5 process/**thread** hangs on soclose IY05185 ...
ww-opensup.bull.com/PTF/packs/ml/aix433/4330011info.html - 232k - Cached - More from this site

ttp://www.freedesktop.org/software/dbus/doc/ChangeLog
ɔus/dbus-string.c (set_length, reallocate_for_length): ignore the test hack that forced constant realloc if asserts a
ɜn profile sanely..
ww.freedesktop.org/software/dbus/doc/ChangeLog - 257k - Cached - More from this site

Results Page:
1 2 3 **Next**

Web | Images | Video | Directory | Local | News | Products

Your Search: "lock contention" unlock stack thread connecting Search

Help us improve your search experience. Send us feedback.

Copyright © 2005 Yahoo! Inc. All rights reserved. Privacy Policy - Terms of Service - Copyright Policy - Submit Your Site - Job Openings

Web Results **1** - **10** of about **150** for **"lock contention" unlock stack thread connecting**. (**0.37** seconds)

[PPT] homepage.cs.uri.edu/courses/csc512/512_Patterns.ppt
File Format: Microsoft Powerpoint 97 - View as HTML
... **Unlock** State info. ... eg, **connecting** with a server on the same host via a 'loopback' device. ... are available & it is efficient to use a **thread**-per-connection to ...
Similar pages

ns_log notice "nsd.tcl: starting to read config file..." ...
... ns/threads ns_param mutexmeter true ;# measure **lock contention** # The per-**thread** **stack** size must be ... MKCOL POST PROPFIND PROPPATCH LOCK **UNLOCK**" #ns_section ns ...
cvs.openacs.org/cvs/openacs-4/etc/config.tcl?rev=1.32 - 27k - Cached - Similar pages

[PDF] Tru64 UNIX 5.1 and TruCluster Server 5.1
File Format: PDF/Adobe Acrobat - View as HTML
Page 1. Tru64 UNIX 5.1 and TruCluster Server 5.1 Patch Summary and Release Notes for Patch Kit-0003 April 2001 This manual describes ...
h30097.www3.hp.com/docs/patch/51/bl17a/ReleaseNotes.pdf - Similar pages

[PDF] ECE398SSL: Computer Systems Engineering Spring 2005 Lecture Notes ...
File Format: PDF/Adobe Acrobat - View as HTML
... next = old head; /* line 4 */ spin **unlock** (&the lock); asm volatile (" # local irq restore macro implementation pushl %0 # save input 0 to **stack** popfl # pop ...
courses.ece.uiuc.edu/ece398/ssl/notes/notes-02.pdf - Similar pages

[PDF] PeopleSoft V8 on zSeries Using Sysplex Data Sharing haring
File Format: PDF/Adobe Acrobat - View as HTML
... PeopleSoft 5 high-speed, duplexed links **connecting** the components. This implementation employs hardware, software, and microcode. ...
www.redbooks.ibm.com/redbooks/pdfs/sg246093.pdf - Similar pages

[PDF] Efficient Synchronization and Coherence for Nonuniform ...
File Format: PDF/Adobe Acrobat - View as HTML
... A lock-**unlock** solutions presented so far, do not ... them are dependent on a per-**thread**/process node_id ... HBO locks and with high **lock contention** (especially Raytrace ...
www.it.uu.se/research/reports/lic/2003-008/2003-008.pdf - Similar pages

Neohapsis Archives - IBM AIX lists - Re: New_AIXV4_Fixes - From ...
... to remove this gap before unlocking the socket. ... Removed unnecessary locking to reduce **lock contention**. ... at 0xd0533b28 The other **stack** is rpc__auth_rpc_prot_epv ...
archives.neohapsis.com/archives/aix/2001-q4/0009.html - 101k - Cached - Similar pages

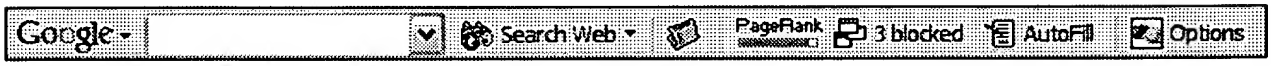
Sybase FAQ: 14/17 - section 10.4
... Note See "Reducing **Lock Contention** with max_rows_per_page ... user mode in Step 2. 2025 Unlocking a Sybase ... additional trace flag, 1205, provided the **stack** trace for ...
www.uni-giessen.de/faq/archiv/ databases.sybase-faq.part1-17.old/msg00013.html - 101k - Cached - Similar pages

[PDF] Hardware and Software Bottlenecks on Large-Scale Shared-Memory ...
File Format: PDF/Adobe Acrobat - View as HTML
... 6.3.1 GAFORT: **Lock Contention** . . ... If multiple threads on each processor access the

same ... Lawrence Livermore Laboratory's Thunder system **connecting** 4096 Intel ...
www-flash.stanford.edu/~rck/thesis.pdf - Similar pages

[PDF] Practical Structures for Parallel Operating Systems
File Format: PDF/Adobe Acrobat - View as HTML
... 5.2.2 The mapinfo System Call 216 5.2.3 **Stack** Growth 216 5.2.4 The fork, spawn ... 4 Ksems 279 7.3.5 Ksem Usage Examples 295 7.4 Lists 310 7.5 **Thread** Management 311 ...
www.cs.nyu.edu/web/Research/Theses/edler_jan.pdf - Similar pages

Free! Get the Google Toolbar. Download Now - About Toolbar



"lock contention" unlock stack threac Search

©2005 Google

CiteSeer Find: lock and unlock and contention and s Documents Citations

Searching for **lock and unlock and contention and stack and thread**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP
4 documents found. **Order: number of citations.**

Using Continuations To Build A User-Level Thread Library - Dean (1993) (Correct) (8 citations)
on a multiprocessor. In addition, flattening the **lock**ing hierarchy reduces context switch latency by 5% in the critical sections. Our multiprocessor **contention** benchmarks indicate that this reduction and the kernel reduced context switching times and kernel **stack** utilization. In C-**Threads**, continuations have not
www.usenix.org/publications/library/proceedings/mach3/full_papers/dean.ps

Experimentation with Configurable, Lightweight Threads.. - Kaushik Ghosh.. (1993) (Correct) (1 citation)
configuration of the **threads** package's mutex **locks** 1 is shown to significantly improve the
ftp.cc.gatech.edu/pub/coc/tech_reports/1993/GIT-CC-93-37.ps.Z

Cost of User and Kernel Level Threads Operations on Linux - Cohen, Patel, Seshagiri (Correct)
the context switch overhead, and the mutex **lock/unlock** overhead. It gives criteria to decide
the context switch overhead, and the mutex **lock/unlock** overhead. It gives criteria to decide whether or
processors and took into account network **contention**, cache interference, context switching
www.eb.uah.edu/~cohen/javanauts/papers/costs2.ps

Exploitation of Multithreading to Improve Program.. - Cohen, Yalamanchilli, .. (1998) (Correct)
the context switch overhead, and the mutex **lock/unlock** overhead. It gives criteria to decide
the context switch overhead, and the mutex **lock/unlock** overhead. It gives criteria to decide whether or
processors and took into account network **contention**, cache interference, context switching
www.eb.uah.edu/~cohen/javanauts/papers/jthreads2.ps

Try your query at: Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP

CiteSeer.IST - Copyright Penn State and NEC